KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

A novel branch-and-bound algorithm for the chance-constrained RCPSP

Davari M, Demeulemeester E.

KBI_1620

A novel branch-and-bound algorithm for the chance-constrained RCPSP

Morteza Davari[†*] and Erik Demeulemeester[†]

Abstract

The resource-constrained project scheduling problem (RCPSP) has been widely studied during the last few decades. In real-world projects, however, not all information is known in advance and uncertainty is an inevitable part of these projects. The chance-constrained resource-constrained project scheduling problem (CC-RCPSP) has been recently introduced to deal with uncertainty in the RCPSP. In this paper, we propose a branch-and-bound (B&B) algorithm and a MILP formulation that solve the CC-RCPSP. We introduce two different branching schemes and eight different priority rules for the proposed B&B algorithm. Since solving CC-RCPSP is computationally intractable, its sample average approximation counterpart is considered to be solved. The computational results suggest that the proposed branch-and-bound procedure clearly outperforms both a proposed MILP formulation and a branch-and-cut algorithm from the literature.
Keywords: Chance-constrained problem; branch-and-bound; CC-RCPSP.

1 Introduction

Real-life projects contain considerable levels of uncertainty. The uncertainty can be the result of many internal and external unexpected or expected events in the execution of a project. This uncertainty may lead to disruptions and may often make any schedule obtained by solving any deterministic project scheduling problem unreliable.

In order to incorporate uncertainty into the *resource-constrained project scheduling problem* (RCPSP) and obtain robust solutions, Lamas and Demeulemeester (2016) introduce the *chance-constrained RCPSP* (CC-RCPSP) in which the makespan is minimized while the actual executed schedule is identical to the planned schedule with a certain minimum probability. In other words, the idea behind CC-RCPSP is to find a proactive solution that is as short as possible and needs no reaction with a certain minimum required

[†]Department of Decision Sciences and Information Management, Faculty of Business and Economics, KU Leuven (Belgium), firstname.lastname@kuleuven.be

[*]Corresponding Author

probability. This probability, which is usually given by the managerial team, will be defined as the *confidence level* in the remainder of this paper.

In this paper, we introduce a novel B&B algorithm that can solve instances of the CC-RCPSP until optimality. The novelty of our B&B algorithm is reflected in its branching schemes. In these branching schemes, we use the notion of *chains* that will be introduced and defined in Section 5.1. Although the focus of this paper is to solve the CC-RCPSP, as will be discussed in Section 7, the proposed branching schemes can be used to solve any *chance-constrained programming* (CCP) problem with discrete random *right-hand side* (rhs) vector provided that an exact solution oracle exists for the problem's deterministic counterpart.

The remainder of this paper is structured as follows. First, in Section 2 a compact description of the CC-RCPSP is given. Then the state of the art around the topic is provided in Section 3. Next, both a *mixed integer linear programming* (MILP) formulation and a B&B algorithm that optimally solve instances of the CC-RCPSP are proposed in Section 4 and Section 5, respectively. After that, computational results are reported in Section 6 and the application of the proposed branching schemes for the general integer problem is discussed in Section 7. Finally, a summary and a conclusion are given in Section 8.

2 Problem description

We are given a set $N = \{0, 1, ..., n+1\}$ of activities where activities 0 and $n+1$ are the dummy start and dummy end activities. Each activity $i \in N' = N \backslash \{0, n+1\}$ has a stochastic non-negative integer duration $\tilde{p}_i$, with $p_i^{\min} \leq \tilde{p}_i \leq p_i^{\max}$, which follows a discrete distribution $dist(\tilde{p}_i)$. We assume that these stochastic durations are independently distributed. Notice that the durations of the dummy activities are not stochastic ($\tilde{p}_0 = \tilde{p}_{n+1} = 0$). We are also given a set $\mathcal{R}$ of renewable resource types. Each job i requires r_{ik} units of resource type $k \in \mathcal{R}$ during its processing time and the resource availability of resource type k is denoted by R_k. The set $E \in \{(i,j)|i,j \in N\}$ defines precedence constraints among the activities where the pair $(i,j) \in E$ indicates that activity j cannot be started before activity i is completed.

Let $\mathbf{S}$, which is a vector of non-negative starting times of the activities, be a solution for the RCPSP and $\bar{\mathbf{p}}$ be its vector of deterministic activity durations. A conceptual formulation for the RCPSP can be formulated as

follows:

$$\text{RCPSP: } \min_{\mathbf{S}} S_{n+1}$$

subject to:

$$S_j - S_i \geq \bar{p}_i \qquad \forall (i,j) \in E \quad (1)$$

$$\sum_{i \in \Gamma_t(\mathbf{S}, \bar{\mathbf{p}})} r_{ik} \leq R_k \qquad \forall k \in \mathcal{R}, t = 0, ..., T \quad (2)$$

$$\mathbf{S} \in \mathbb{N}^{n+2} \quad (3)$$

where T is an upper bound on the makespan and $\Gamma_t(\mathbf{S}, \bar{\mathbf{p}})$ is the corresponding set of ongoing activities at time period $(t, t+1)$ if $\mathbf{S}$ is the vector of starting times of the activities and $\bar{\mathbf{p}}$ represents the activity durations. In the above formulation, the objective function is to minimize the completion time of the project (makespan). Constraints (1) ensure that all precedence relations among activities are fulfilled, whereas constraints (2) represent the resource constraints.

Let $\pi(\cdot)$ be the probability that constraints $\cdot$ are satisfied and $(1-\alpha)$ be the confidence level defined by the decision maker (note that $0 \leq \alpha \leq 1$ and α usually takes a value very close to 0, for example $\alpha = 0.05$, $\alpha = 0.10$ or $\alpha = 0.20$, otherwise the resulting schedule is not feasible for a large number of cases)[1]. A conceptual formulation for the CC-RCPSP is given as follows:

$$\text{CC-RCPSP: } \min_{\mathbf{S}} S_{n+1}$$

subject to constraint (3) and

$$\pi \left(\begin{array}{ll} S_j - S_i \geq \tilde{p}_i & \forall (i,j) \in E \\ \sum_{i \in \Gamma_t(\mathbf{S}, \tilde{\mathbf{p}})} r_{ik} \leq R_k & \forall k \in \mathcal{R}, t = 0, ..., T \end{array} \right) \geq 1 - \alpha \quad (4)$$

In the above formulation, constraint (4) ensures that the sets of constraints (1) and (2) combined are not violated with a chance of $(1-\alpha)$. The CC-RCPSP is proven to be strongly NP-hard following the straightforward reduction from the RCPSP.

2.1 A realization-based reformulation

The vector $\tilde{\mathbf{p}} = (\tilde{p}_0, \tilde{p}_1, ..., \tilde{p}_{n+1})$ can be represented by a finite supporting set $\mathcal{P} = \{\mathbf{p}^1, ..., \mathbf{p}^{|\mathbf{P}|}\}$ of realizations where each realization $\mathbf{p}^l$ represents a vector of processing times $\mathbf{p}^l = (p_0^l, p_1^l, ..., p_{n+1}^l) \in \mathcal{P}$. Each realization $\mathbf{p}^l$ occurs

[1] All probability operators in this paper adhere to the Kolmogorov's Axioms.

with a certain probability π_l. Clearly, the summation of the probabilities of all realizations equals one ($\sum_{\mathbf{p}^l \in \mathcal{P}} \pi_l = 1$).

Being a probabilistic constraint, Constraint (4) is very difficult to tackle. Alternatively, we decide to ensure the feasibility of each solution by finding a *sufficient* subset Y of realizations for which the solution is feasible. A subset Y of realizations is called sufficient if and only if $\sum_{\mathbf{p}^l \in Y} \pi_l \geq (1-\alpha)$. We introduce $\mathbf{p}^Y$ as the vector of maximum durations of subset Y which is computed as follows:

$$p_i^Y = \max_{\mathbf{p}^l \in Y} \{p_i^l\} \qquad \forall i \in N \tag{5}$$

Let $\mathcal{Y}_\mathcal{P}$ be the set of all sufficient subsets of $\mathcal{P}$. The CC-RCPSP can be reformulated as follows:

$$\text{CC-RCPSP-R} : \min_{(Y,\mathbf{S})} S_{n+1}$$

subject to constraint (3) and

$$S_j - S_i \geq p_i^Y \qquad \forall (i,j) \in E \tag{6}$$

$$\sum_{i \in \Gamma_t(\mathbf{S},\mathbf{p}^Y)} r_{ik} \leq R_k \qquad \forall k \in \mathcal{R}, t = 0,...,T \tag{7}$$

$$\sum_{\mathbf{p}^l \in Y} \pi_l \geq (1-\alpha) \tag{8}$$

$$Y \in \mathcal{Y}_\mathcal{P} \tag{9}$$

In the above formulation, constraints (6) make sure that no precedence violation occurs for any realization $\mathbf{p}^l \in Y$ and constraints (7) ensure that no resource violation occurs for any realization $\mathbf{p}^l \in Y$. Constraint (8) dictates a confidence level of $(1-\alpha)$. In the special case where $\pi_l = 1/m$ for all realizations $\mathbf{p}^l$ in $\mathcal{P}$, constraint (8) can be replaced by constraint (10).

$$|Y| \geq \lceil (1-\alpha) \times m \rceil \tag{10}$$

2.2 A sample average approximation

The size of the associated finite supporting set of realizations is often too large, and thus we use a *sample average approximation* (SAA) technique, which is based on Monto Carlo sampling, to generate a much smaller set $\hat{\mathcal{P}}$ of m realizations which approximates the original set $\mathcal{P}$ (note that the size m influences the quality of the approximation).

We introduce the associated SAA counterpart of the CC-RCPSP (in short SAA-RCPSP) as follows:

$$\text{SAA-RCPSP}: \min_{(Y,\mathbf{S})} S_{n+1}$$

subject to constraint (3),(6),(7) and

$$\sum_{\mathbf{p}^l \in Y} \pi_l \geq (1-\hat{\alpha}) \tag{11}$$

$$Y \in \mathcal{Y}_{\hat{\mathcal{P}}} \tag{12}$$

where $1-\hat{\alpha}$ is the required confidence level for this counterpart problem (note that $1-\hat{\alpha} > 1-\alpha$).

Let $\epsilon = \alpha - \hat{\alpha}$ and let $p^{\max} = \max_{i \in N}\{p_i^{\max}\}$. Following the theorems in Luedtke and Ahmed (2008), one can show that any feasible solution to an instance of SAA-RCPSP is also feasible to the associated instance of CC-RCPSP with a probability of $(1-\theta)$ if

$$m \geq \frac{1}{2\epsilon^2} \log\left(\frac{1}{\theta}\right) + \frac{n}{2\epsilon^2} \log(p^{\max}). \tag{13}$$

For instance if $p^{\max} = 30$, then m must be at least 9123 to ensure that any feasible solution to an instance of SAA-RCPSP is also feasible to the associated instance of CC-RCPSP with a probability of 0.95. However, Luedtke and Ahmed (2008) also argue that this lower bound for m is very conservative and one often achieves similar confidence with much smaller m. In this paper, we choose m between 100 and 1600.

As the final part of this section, we pinpoint two remarks. First, because solving CC-RCPSP becomes computationally intractable, in Sections 4 and 5, we opt to solve the SAA-RCPSP, which is proven to be a good approximation for CC-RCPSP (Lamas and Demeulemeester, 2016). However, given unlimited computational resources, the methods presented in Sections 4 and 5 can solve instances of CC-RCPSP. Second, while using the SAA technique to generate a set of realizations, generally the probability of occurrence of every single generated realization is the same as that of any other realization in that set and equals $1/m$. However, our methods are designed for a more general case where the probabilities of occurrences of realizations need not be the same.

3 Literature review

In this section, we review the state of the art around the topic of the CC-RCPSP in three parts. In the first part (Section 3.1), the literature on the deterministic RCPSP is presented. In the second part (Section 3.2), we list a number of papers dealing with the RCPSP under uncertainty. And finally in the last part (Section 3.3), a brief review of the literature on chance-constrained programming is given.

3.1 The deterministic RCPSP

The deterministic RCPSP is known to be NP-hard in the strong sense (Blazewicz et al., 1983). However, a number of relatively efficient MILP formulations as well as other exact methods, such as B&B and *branch-and-cut* (B&C) algorithms, have been presented to solve this NP-hard problem (Demeulemeester and Herroelen, 2002). Many MILP formulations for the deterministic RCPSP have been proposed. Among these, we refer to Klein (2000) who gathers six different MILP formulations for the deterministic RCPSP. Also recently, other MILP formulations have been introduced, for which we refer to Artigues et al. (2008) and Koné et al. (2011). Among many exact algorithms that are proposed during the last few decades, B&B algorithms (Demeulemeester and Herroelen, 1992, 1996, 1997; Klein, 2000; Mingozzi et al., 1998; Sprecher, 2000) are the most successful approaches to optimally solve small and medium-sized instances of the deterministic RCPSP.

3.2 The RCPSP under uncertainty

As already motivated in the introduction, handling uncertainty in project scheduling is of great importance. The CC-RCPSP, which is the main topic of this paper and which is described in Section 2, is one of the existing problems that incorporates uncertainty in project scheduling. Besides CC-RCPSP, there are two other main streams of research dealing with RCPSP in the presence of activity duration uncertainty: the stochastic RCPSP and the proactive and reactive resource-constrained project scheduling problem.

The first trend in the literature focuses on the *stochastic RCPSP* which is also referred to as SRCPSP. The SRCPSP considers stochastic activity durations while minimizing the makespan. A solution in the SRCPSP is a *policy* which is a set of decision rules that dictate the starting of certain activities at certain decision moments in a dynamic fashion (Stork, 2000).

The second trend in the literature focuses on the *proactive and reactive resource-constrained project scheduling problem* (PR-RCPSP). The idea of proactive and reactive scheduling is to generate a baseline schedule that is as robust as possible against disruptions (which is referred to as *proactive scheduling*) and to propose a reactive policy that determines how to react to possible disruptions (which is referred to as *reactive scheduling*) (Van de Vonder et al., 2006b).

The SRCPSP is already known for many decades. It all started when Möhring and Radermacher (1985) provided an introduction for stochastic project scheduling problems and Möhring et al. (1984, 1985) investigated the analytical aspects of these problems and provided analytical discussions and proofs. Since finding a globally optimal policy was not tractable except until recently (Creemers, 2015), a large body of research is devoted on searching over different classes of policies (Ashtiani et al., 2011; Möhring et al., 1984, 1985; Rostami et al., 2016; Stork, 2001).

A very crucial disadvantage of the SRCPSP is that no baseline schedule is constructed. This disadvantage of the SRCPSP was the trigger for the initial steps towards PR-RCPSP in which a stable baseline schedule is generated. A number of justifications for the necessity of a baseline schedule have been discussed by several authors such as Aytug et al. (2005), Demeulemeester and Herroelen (2011), Herroelen (2005), Leus (2003) and Mehta and Uzsoy (1998). Among many research papers that study the PR-RCPSP, we cite the recent papers by Davari and Demeulemeester (2016), Deblaere et al. (2011a,b), Herroelen and Leus (2004), Leus and Herroelen (2005) and Van de Vonder et al. (2006a, 2008).

3.3 Chance-constrained programming

To the best of our knowledge, the first research on CCP problems with discrete random rhs vector is the one studied by Prékopa (1990). Ruszczyński (2002) proposes a mixed integer programming formulation together with some valid inequalities for a CCP problem with a rhs vector. Recently, stronger facet-defining valid inequalities are proposed by Luedtke et al. (2010) and Küçükyavuz (2012). Among many papers in which a CCP problem with discrete random rhs vector has been introduced to handle uncertainty, we only cite Lamas and Demeulemeester (2016) who introduce the CC-RCPSP and propose a B&C algorithm that solves instances of its SAA counterpart (SAA-RCPSP) to optimality.

4 A mathematical formulation

In this section, we introduce a MILP formulation for the SAA-RCPSP which is the chance-constrained version of the resource-flow formulation proposed by Artigues et al. (2003). Let's define variables x_{ij} that equal one if activity i is completed before the start of activity j and zero otherwise. We also introduce the variables f_{ijk} which represent the amount of resource type k that is passed from activity i to activity j. Finally, we define variables y_l such that if $y_l = 1$, then S must be feasible for realization $\mathbf{p}^l$ ($\mathbf{p}^l \in Y$) and if $y_l = 0$, then the feasibility of S for realization $\mathbf{p}^l$ is not necessary ($\mathbf{p}^l \notin Y$). We propose the following MILP formulation for the SAA-RCPSP:

$$\text{SAA-RCPSP-MILP} : \min_{\mathbf{S}} S_{n+1}$$

subject to:

$$x_{ij} = 1 \qquad \forall (i,j) \in E \tag{14}$$

$$x_{ij} + x_{ji} \leq 1 \qquad \forall (i,j) \in N^2, i \neq j \tag{15}$$

$$f_{ijk} \leq M^r x_{ij} \qquad \forall (i,j) \in N'^2, i \neq j, \forall k \in \mathcal{R} \tag{16}$$

$$f_{0jk} \leq r_{jk} x_{0j} \qquad \forall (0,j) \in N'^2, j \neq 0, \forall k \in \mathcal{R} \tag{17}$$

$$f_{i(n+1)k} \leq r_{ik} x_{i(n+1)} \qquad \forall (i,n+1) \in N'^2, i \neq n+1, \forall k \in \mathcal{R} \tag{18}$$

$$\sum_{j \in N} f_{0jk} = R_k \qquad \forall k \in \mathcal{R} \tag{19}$$

$$\sum_{j \in N} f_{ijk} = r_{ik} \qquad \forall i \in N, \forall k \in \mathcal{R} \tag{20}$$

$$\sum_{j \in N} f_{jik} = r_{ik} \qquad \forall i \in N, \forall k \in \mathcal{R} \tag{21}$$

$$S_j - S_i - M^p(x_{ij} - 1) \geq y_l p_i^l \qquad \forall (i,j) \in N^2, i \neq j, \forall l = 1, ..., m \tag{22}$$

$$\sum_{l=1}^{m} (1 - y_l)\pi_l \leq \hat{\alpha} \tag{23}$$

$$\mathbf{S} \in \mathbb{N}^{n+2}, \mathbf{x} \in \{0,1\}^{(n+2)^2}, \tag{24}$$

$$\mathbf{f} \in \mathbb{R}^{|\mathcal{R}|(n+2)^2}, \mathbf{y} \in \{0,1\}^m \tag{25}$$

where $M^p = \sum_{i \in N} p_i^{\max}$ and $M^r = \min\{r_{ik}, r_{jk}\}$. In the above formulation, the set of constraints (14) enforces the precedence relations among activities. The set of constraints (15) ensures that the two variables x_{ij} or x_{ji} are not both 1. The sets of constraints (16)-(21) represent the resource flow among activities. Constraints (22) guarantee that $\mathbf{S}$ is feasible for realization $\mathbf{p}^l$ if

$y_l = 1$. Finally, constraint (23) ensures that the cumulative probability of the occurrences of the realizations in the set Y (realizations $\mathbf{p}^l$ for which $y_l = 1$) must be at least $(1 - \hat{\alpha})$. Based on the definitions, for each feasible vector $\mathbf{y}$, we can derive an associated sufficient set of realizations Y. Let us introduce $\mathbf{p}^{\mathbf{y}}$ which equals its associated vector $\mathbf{p}^Y$. We have:

$$p_i^Y = p_i^{\mathbf{y}} = \max_{\mathbf{p}^l \in \hat{\mathcal{P}}} \{y_l p_i^l\} = \max_{\mathbf{p}^l \in Y} \{p_i^l\}.$$

The set of constraints (22) can be replaced with the following set of conceptual constraints:

$$S_j - S_i - M^p(x_{ij} - 1) \geq p_i^{\mathbf{y}} \qquad \forall (i,j) \in N^2, i \neq j \quad (26)$$

Although constraints (26) form a nonlinear term, they can be used to understand which realizations should be included in or excluded from Y. An obvious finding is that in order to reduce the makespan (S_{n+1}) and possibly find an optimal solution, we must choose $\mathbf{y}$ in such a way that for some activities i, the value $p_i^{\mathbf{y}}$ takes a smaller value than $p_i^{\max}$. This finding is the main motivation for the reformulation introduced in the following subsection.

4.1 A stronger formulation

In Luedtke et al. (2010), a stronger reformulation has been proposed for the general chance-constrained optimization problem. We follow the steps to obtain a stronger reformulation for the SAA-RCPSP. Let δ_i be the vector of activity durations for each realization in $\hat{\mathcal{P}}$, sorted in non-increasing order. In other words, δ_i^k is the k^{th} largest duration for activity i. In order to keep track of the realizations in each sorted vector δ_i, we introduce σ_{ik} which represents the associated realization for each pair (i,k) (in other words, we have $\delta_i^k = p_i^{\sigma_{ik}}$). For example, if p_i^5 is the second largest duration for activity i, then $\delta_i^2 = p_i^5$ and $\sigma_{i2} = 5$. We provide a more detailed example in Section 4.2. For each activity i, we define $\eta_i \leq m$ as the largest $k \in \{1, ..., m\}$ that satisfies $\sum_{s=1}^{k} \pi_{\sigma_{is}} \leq \hat{\alpha}$. Parameter η_i can be described as an upper bound on the number of realizations $\mathbf{p}^l$ that are necessary to be excluded from Y ($y_l = 0$) to achieve the minimum $p_i^{\mathbf{y}}$. Notice that $\eta_0 = \eta_{n+1} = 0$.

Thus, a stronger formulation can be obtained for the SAA-RCPSP:

SAA-RCPSP-MILP2: $\min_{\mathbf{S}} S_{n+1}$

subject to: constraints (14)-(21), (23)-(25) and

$$S_j - S_i - M^p(x_{ij} - 1) \geq \delta_i^1 - \sum_{k=1}^{\eta_i} (\delta_i^k - \delta_i^{k+1})(1 - z_{ik}) \qquad \forall (i,j) \in N^2, i \neq j \tag{27}$$

$$z_{ik} - y_{\sigma_{ik}} \geq 0 \qquad \forall i \in N, k = 1, ..., \eta_i \tag{28}$$

$$z_{i,k+1} - z_{ik} \geq 0 \qquad \forall i \in N, k = 1, ..., \eta_i \tag{29}$$

$$z_{i,\eta_i+1} = 1 \qquad \forall i \in N \tag{30}$$

$$z_{ik} \in \{0, 1\} \qquad \forall i \in N, k = 1, ..., \eta_i \tag{31}$$

In the above formulation, z_{ik} is zero if we decide not to consider the k^{th} largest duration for activity i and one otherwise. Also if $z_{ik} = 0$, then the feasibility of the resulting schedule is not guaranteed for its associated realization (σ_{ik}) and thus we enforce $y_{\sigma_{ik}} = 0$ (constraints (28)). Obviously, it would be inefficient if $z_{ik} = 1$ and $z_{i,k+1} = 0$. Therefore, such cases are eliminated by constraints (29). It is not very difficult to see that in no feasible schedule z_{i,η_i+1} could be equal to zero and therefore the strength of the linear relaxation bound can be improved by adding constraints (30). Notice that the set of constraints (27)-(31) is a much stronger alternative for the set of constraints (22).

4.2 An example

We consider an instance of the problem with $n = 8$ and $m = 10$. The precedence relations among activities as well as the resource consumptions are given in Figure 1. An example set of realizations $\hat{\mathcal{P}}$ and the associated matrices δ and σ are given in Tables 1 to 3.

In these three tables, the numbers associated with realization $\mathbf{p}^6$ are shown in bold. Note that these tables will be used in all examples given in the remainder of this paper.

5 Branch-and-bound

In this section, we propose a B&B algorithm that solves the SAA-RCPSP. The idea is to find a schedule with minimum makespan that is feasible for

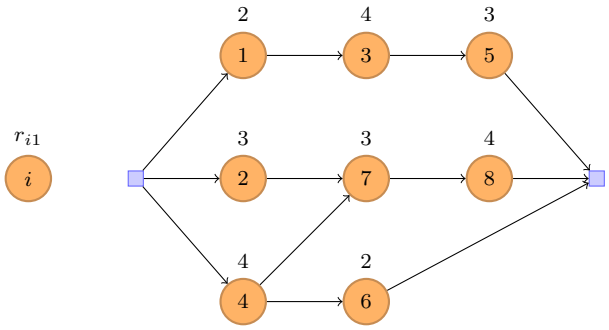

Figure 1: The graph of precedence relations among activities

Table 1 The set $\hat{\mathcal{P}}$ of realization for the example. Each column represents a realization

i	p_i^1	p_i^2	p_i^3	p_i^4	p_i^5	p_i^6	p_i^7	p_i^8	p_i^9	p_i^{10}
1	3	3	3	2	2	**3**	3	1	1	3
2	10	5	9	6	8	**6**	7	11	11	6
3	2	3	4	3	3	**2**	5	4	5	1
4	4	3	6	6	6	**5**	4	2	4	4
5	7	7	5	9	12	**6**	6	5	9	9
6	7	9	4	6	5	**4**	9	7	9	8
7	3	4	4	2	5	**4**	6	4	6	2
8	1	1	3	3	2	**3**	3	3	2	2

Table 2 The matrix δ for the example

i	δ_i^1	δ_i^2	δ_i^3	δ_i^4	δ_i^5	δ_i^6	δ_i^7	δ_i^8	δ_i^9	δ_i^{10}
1	3	3	3	**3**	3	3	2	2	1	1
2	11	11	10	9	8	7	6	**6**	6	5
3	5	5	4	4	3	3	3	2	**2**	1
4	6	6	6	**5**	4	4	4	4	3	2
5	12	9	9	9	7	7	**6**	6	5	5
6	9	9	9	8	7	7	6	5	4	**4**
7	6	6	5	4	4	**4**	4	3	2	2
8	3	3	**3**	3	3	2	2	2	1	1

at least one sufficient set of realizations. Let Y denote a set of realizations and let $\mathcal{O}(\mathbf{p}^Y)$ be an optimization oracle that solves the RCPSP while the vector of activity durations is $\mathbf{p}^Y$. We define S^Y as the schedule obtained by running $\mathcal{O}(\mathbf{p}^Y)$. Because S^Y is feasible for the problem with $\mathbf{p}^Y$, it is

Table 3 The matrix σ for the example

i	$\sigma_{i,1}$	$\sigma_{i,2}$	$\sigma_{i,3}$	$\sigma_{i,4}$	$\sigma_{i,5}$	$\sigma_{i,6}$	$\sigma_{i,7}$	$\sigma_{i,8}$	$\sigma_{i,9}$	$\sigma_{i,10}$
1	1	2	3	**6**	7	10	4	5	8	9
2	8	9	1	3	5	7	4	**6**	10	2
3	7	9	3	8	2	4	5	1	**6**	10
4	3	4	5	**6**	1	7	9	10	2	8
5	5	4	9	10	1	2	**6**	7	3	8
6	2	7	9	10	1	8	4	5	3	**6**
7	7	9	5	2	3	**6**	8	1	4	10
8	3	4	**6**	7	8	5	9	10	1	2

also feasible for all realizations in Y. Thus, if Y is a sufficient subset, then it provides a confidence level of $(1 - \hat{\alpha})$.

For each Y, there exists a complement set $\bar{Y}$ of realizations such that $Y \cup \bar{Y} = \hat{\mathcal{P}}$. For each pair $(Y, \bar{Y})$, Y is referred to as the *included set* and $\bar{Y}$ is referred to as the *excluded set*. Obviously, if Y is a sufficient subset, then the cumulative probability of occurrence of the realizations in $\bar{Y}$ is smaller than or equal to $\hat{\alpha}$. Let Ξ be the set of all pairs $(Y, \bar{Y})$ for which $\sum_{\mathbf{p}^l \in \bar{Y}} \pi_l \leq \hat{\alpha}$. The SAA-RCPSP can be reformulated as follows:

$$\min_{(Y,\bar{Y}) \in \Xi} S_{n+1}^{Y} \tag{32}$$

Inspired by this conceptual formulation, we aim to use a B&B algorithm to enumerate all pairs $(Y, \bar{Y}) \in \Xi$ and find a pair $(Y^*, \bar{Y}^*)$ with the minimum corresponding makespan $(S_{n+1}^{Y^*})$.

5.1 Constructing the tree

As we already mentioned, we aim to find the optimal pair $(Y^*, \bar{Y}^*)$ using a B&B algorithm. Since the included set (Y) and the excluded set $(\bar{Y})$ are the complement of each other, it is sufficient to only enumerate all valid excluded sets (or all valid included sets). A conventional branching scheme can be perfectly used to enumerate all excluded sets by starting from the set of all realizations and in each node/level excluding a single realization. However, we opt not to directly exclude single realizations, but instead we exclude chains (sets) of realizations in our novel branching schemes.

Let us introduce C_i^k as the *chain* of realizations with the k^{th} highest duration for activity i. The exclusion of chain C_i^k from pair $(Y, \bar{Y}) \in \Xi$

results in the pair $(Y \setminus C_i^k, \bar{Y} \cup C_i^k)$. This exclusion is *possible* only if the resulting pair $(Y \setminus C_i^k, \bar{Y} \cup C_i^k)$ is also a member of Ξ.

We immediately notice that not all possible exclusions are necessary to be evaluated. Only those exclusions that have a positive impact must be considered. Therefore, we introduce the notion of *beneficial* exclusions. The exclusion of chain C_i^k from pair $(Y, \bar{Y}) \in \Xi$ is labeled beneficial if and only if it is possible and $\mathbf{p}^{Y \setminus C_i^k} < \mathbf{p}^Y$ (Obviously if $\mathbf{p}^{Y \setminus C_i^k} = \mathbf{p}^Y$, then $S^{Y \setminus C_i^k} = S^Y$ and the associated exclusion is not beneficial).

Corollary 1. *The exclusion of chain C_i^k from pair $(Y, \bar{Y}) \in \Xi$ is beneficial if it is possible, $\bar{Y} \cup C_i^k \neq \bar{Y}$ and $(C_i^1 \cup ... \cup C_i^{k-1}) \subseteq \bar{Y}$. The reverse relation does not necessarily hold.*

Some chains can never be beneficially excluded. Since considering such chains is not efficient, we limit our search to only *eligible* chains.

Definition (Eligible chain). An eligible chain is a chain that can be beneficially excluded.

We introduce set $\mathbf{C}^E$ as the set of all eligible chains. Let us compute $\pi(C_i^k) = \sum_{\mathbf{p}^l \in C_i^k} \pi_l$. The following proposition is derived.

Proposition 1. *A chain C_i^k is eligible if and only if $\sum_{s=1}^k \pi(C_i^s) \leq \hat{\alpha}$.*

Proof: Consider sets $Y_1 = \hat{\mathcal{P}} \setminus (C_i^1 \cup ... \cup C_i^{k-1})$ and $Y_2 = Y_1 \setminus C_i^k$. If the inequality $\sum_{s=1}^k \pi(C_i^s) \leq \hat{\alpha}$ holds, both sets are sufficient sets. We also know that $p_i^{Y_2} < p_i^{Y_1}$. Therefore, we conclude that the exclusion of C_i^k from Y_1 is beneficial and C_i^k is eligible.

On the other hand, if C_i^k is eligible, it can be beneficially excluded and therefore $(C_i^1 \cup ... \cup C_i^{k-1}) \subseteq Y$. Since any beneficial exclusion is possible, we have $\sum_{s=1}^k \pi(C_i^s) \leq \hat{\alpha}$. □

We define ζ_i as the number of eligible chains associated with activity i. For each activity i, the chain corresponding to the highest duration (C_i^1) is referred to as the *lead chain*. Other chains that associate with the second duration, third duration, etc are referred to as the *second chain, third chain, etc*, respectively.

Example. *Let $\hat{\alpha} = 0.4$ and*

$$\pi = (\pi_1, ..., \pi_{10}) = (0.2, 0.15, 0.15, 0.1, 0.1, 0.1, 0.05, 0.05, 0.05, 0.05).$$

The set of eligible chains for the example is

$$\begin{aligned}\mathbf{C}^E = \{&C_2^1 = \{8,9\}, C_2^2 = \{1\},\\ &C_3^1 = \{7,9\}, C_3^2 = \{3,8\},\\ &C_4^1 = \{3,4,5\},\\ &C_5^1 = \{5\}, C_5^2 = \{4,9,10\},\\ &C_6^1 = \{2,7,9\}, C_6^2 = \{10\},\\ &C_7^1 = \{7,9\}, C_7^2 = \{5\}\}\end{aligned}$$

In this case, the lead chains are $C_2^1, C_3^1, C_4^1, C_5^1, C_6^1$ *and* C_7^1*. We also compute:* $\zeta_0 = \zeta_1 = \zeta_8 = \zeta_9 = 0$*,* $\zeta_4 = 1$ *and* $\zeta_2 = \zeta_3 = \zeta_5 = \zeta_6 = \zeta_7 = 2$*.*

5.1.1 Constructing the activity list

In this part, we construct an *activity list* (AL) (note that all activities i for which $\zeta_i = 0$ are not considered in this) that defines the order based on which activities are considered in our proposed branching schemes (that are introduced in Sections 5.1.2 and 5.1.3). We introduce different *priority rules* that can be used to construct such an AL. Each priority rule consists of a *sorting criterion* based on which the activities are sorted and (possibly) a number of tie breakers. Two example priority rules are given below.

(R_1) We sort activities based on the lexicographical order of the realizations in their σ_i vector.

(R_2) We sort activities based on the decreasing order of their lead chain sizes. As the first/second/etc tie breaking rule, we consider the sizes of their second/third/etc chains.

Although the performance of these two priority rules are acceptable (see Section 6.2), better rules can be achieved by incorporating more important criteria. We propose to take three criteria into consideration while sorting the activities:

- The first criterion is the *total slack* (TS) of the activity. Given realization $\mathbf{p}$ and a feasible schedule S, let $es_i(S,\mathbf{p})$ and $ls_i(S,\mathbf{p})$ be the earliest and the latest starting times of activity i, respectively. We denote by $\varepsilon_i(S,\mathbf{p})$ the total slack of activity i for the given pair $(S,\mathbf{p})$. This total slack is computed as follows: $\varepsilon_i(S,\mathbf{p}) = ls_i(S,\mathbf{p}) - es_i(S,\mathbf{p})$.

Activities with smaller total slack values are favored to be positioned first in the list. In our experiments, we use $S^{\mathbf{p}^{\max}}$ and $\mathbf{p}^{\max}$ to compute total slacks.

- The second criterion is the number of chains (NC) associated to the activity. Those activities with smaller numbers of chains (smaller ζ_i) are favored to be positioned first in the list.

- The last criterion is the *influence factor* (IF) of the activity. The influence factor of activity i, which is denoted by ϑ_i, is computed as follows:

$$\vartheta_i = \sum_{k=1}^{\zeta_i} \frac{p_i^{\max} - v(C_i^k)}{\sum_{s=1}^{k} |C_i^s|}$$

where $v(C_i^k)$ represents the resulting duration of activity i if C_i^k and its corresponding earlier chains (C_i^{k-1},C_i^{k-2},etc) are eliminated. Activities with larger influence factor values are favored to be positioned first in the list.

Example. *We compute the IF for activity* 5 *as follows:*

$$\vartheta_5 = \frac{12-9}{1} + \frac{12-7}{4} = 4.25$$

One of these three criteria is selected as the main sorting criterion. The other two criteria are exploited as the first and the second tie breakers. The question is in which order we need to consider these three criteria such that the performance of the branch-and-bound algorithm is maximized. These criteria can be ordered in six different ways Table 4 depicts these six different ways, each associated with a different priority rule $(R_3, ..., R_8)$. Notice that all priority rules $(R_1 - R_8)$ share a final tie breaking rule which dictates the activity with the smaller index to be positioned first.

5.1.2 Branching scheme 1

The nodes in our B&B are denoted by $\mathcal{N}_s$ where s is the index of the node, indicating the sequence in which the nodes are visited. In each node $\mathcal{N}_s$ (except in the root node), we decide to exclude a chain $C_i^k \in \mathbf{C}^E$ that is referred to as the *target chain*. The target chain of node $\mathcal{N}_s$ is denoted by $\Theta(\mathcal{N}_s)$. The *direct father* of a node $\mathcal{N}_s$ is the node from which it branched whereas a node's *transitive father* is an ancestor (*i.e.* father of the father

Table 4 Different priority rules obtained by different combinations of the following three criteria: total slack (TS), number of chains (NC) and influence factor (IF). Symbol (↑) represents an ascending order and (↓) denotes a descending order.

Priority rule	Sorting criterion	First tie breaker	Second tie breaker
R_3	TS (↑)	NC (↑)	IF (↓)
R_4	TS (↑)	IF (↓)	NC (↑)
R_5	NC (↑)	TS (↑)	IF (↓)
R_6	NC (↑)	IF (↓)	TS (↑)
R_7	IF (↓)	TS (↑)	NC (↑)
R_8	IF (↓)	NC (↑)	TS (↑)

(grandfather), father of the grandfather, etc.) of the node. A node's set of excluded chains is the set of all target chains of the node, its direct father and all of its transitive fathers. Since each node has a one-to-one correspondence with its set of excluded chains, without loss of generality, we let both the node and its associated set of excluded chains be represented by the same notation $\mathcal{N}_s$. Excluding a chain is equivalent to the exclusion of all its realizations. For each node $\mathcal{N}_s$, the pair $(Y^{\mathcal{N}_s}, \bar{Y}^{\mathcal{N}_s})$ is the node's associated pair in Ξ. Notice that a chain may be considered excluded, before being excluded itself, with the exclusion of a combination of some other chains.

Example. *For this example,* $\mathcal{N}_0 = \emptyset$ *represents the root node,* $\mathcal{N}_1 = \{C_6^1\}$*, which is branched from* $\mathcal{N}_0$*, is the node where only* C_6^1 *is excluded and* $\mathcal{N}_2 = \{C_6^1, C_6^2\}$*, which is branched from* $\mathcal{N}_1$*, represents the node where both* C_6^1 *and* C_6^2 *are excluded. The root node is the father of* $\mathcal{N}_1$ *and the only transitive father of* $\mathcal{N}_2$*. The target chains are* $\Theta(\mathcal{N}_1) = C_6^1$ *and* $\Theta(\mathcal{N}_2) = C_6^2$ *for* $\mathcal{N}_1$ *and* $\mathcal{N}_2$*, respectively. Also,* $Y^{\mathcal{N}_2} = \{\mathbf{p}^1, \mathbf{p}^3, \mathbf{p}^4, \mathbf{p}^5, \mathbf{p}^6, \mathbf{p}^8\}$ *and* $\bar{Y}^{\mathcal{N}_2} = \{\mathbf{p}^2, \mathbf{p}^7, \mathbf{p}^9, \mathbf{p}^{10}\}$.

Each node $\mathcal{N}_s$ corresponds with a set $D(\mathcal{N}_s)$ of *effective chains*. For $\mathcal{N}_s$, a chain C is an effective chain if its exclusion from $\mathcal{N}_s$ is beneficial and its associated activity is positioned after the associated activity of $\Theta(\mathcal{N}_s)$. The former condition guarantees an improvement in the child's vector of durations whereas the latter condition prevents duplicate exclusions of chains. Since it is not efficient to exclude any ineffective chain, the target chains of the children of a node must be members of its set of effective chains.

Example. *The set of effective chains for the root node and for priority rule* R_1 *consists of all lead chains (*$D(\mathcal{N}_0) = \{C_6^1, C_4^1, C_5^1, C_3^1, C_7^1, C_2^1\}$*) and*

therefore no child of the root node has a non-lead target chain. For the node $\mathcal{N}_2 = \{C_6^1, C_6^2\}$ *we have:*

$$D(\mathcal{N}_2) = \{C_5^1, C_7^2, C_2^1\}$$

Note that C_3^1 *and* C_7^1 *are both subsets of* $\bar{Y}^{\mathcal{N}_2}$ *and thus the exclusions of* C_3^1 *and* C_7^1 *from* $(Y^{\mathcal{N}_2}, \bar{Y}^{\mathcal{N}_2})$ *are not beneficial. Additionally, the exclusions of* C_3^2 *and* C_4^1 *from* $(Y^{\mathcal{N}_2}, \bar{Y}^{\mathcal{N}_2})$ *are not possible. Since all effective chains must be both beneficial and possible,* $D(\mathcal{N}_2)$ *only consists of* C_5^1, C_7^2 *and* C_2^1*. With similar considerations, we have:* $D(\mathcal{N}_4) = D(\mathcal{N}_5) = D(\mathcal{N}_6) = \emptyset$*.*

The branching starts with the *root* node ($\mathcal{N}_0$). The root node, which corresponds with the situation where no chain has been excluded ($\mathcal{N}_0 = \emptyset$), is branched into a number of child nodes, each corresponding with the exclusion of a certain chain (remember that this chain must be a member of $D(\mathcal{N}_0)$ and therefore should be both possible and beneficial). Each of these child nodes is then branched into its own children and so on. Backtracking happens in a node if all its children have already been visited or if its set of eligible chains is an empty set. Each node in this B&B tree is associated with a feasible solution for SAA-RCPSP. Thus, $\text{UB}^{\mathcal{N}_s} = S_{n+1}^{Y^{\mathcal{N}_s}}$ is an upper bound for the SAA-RCPSP. We denote the best upper bound found so far by UB^*.

Although we avoid duplicated combinations of excluded chains by introducing the set of effective chains, it is very difficult to modify the branching scheme such that no duplicated combination of excluded realizations occurs. This difficulty stems from the fact that many chains may contain one or more common realizations. In order to clarify this branching scheme, we provide an example in which the B&B tree is searched in depth-first mode.

Example. *Figure 2 depicts a part of the B&B tree where branching scheme 1 is used in a depth-first mode. Each node is represented by a square. Since all information of a chain cannot be printed for each node (because of limited space), only its target chain is printed. Also, due to lack of space, the tree is not complete (the nodes with white background have not been continued and the nodes with colored background have been continued).*

The root node is branched into nodes with effective target chains $C_6^1, C_4^1, C_5^1, C_3^1, C_7^1$ *and* C_2^1*. Among the children of the root node, node* $\mathcal{N}_1 = \{C_6^1\}$ *is branched first since its target chain's associated activity is positioned earlier in the AL (which is constructed according priority rule* R_1 *for this example).*

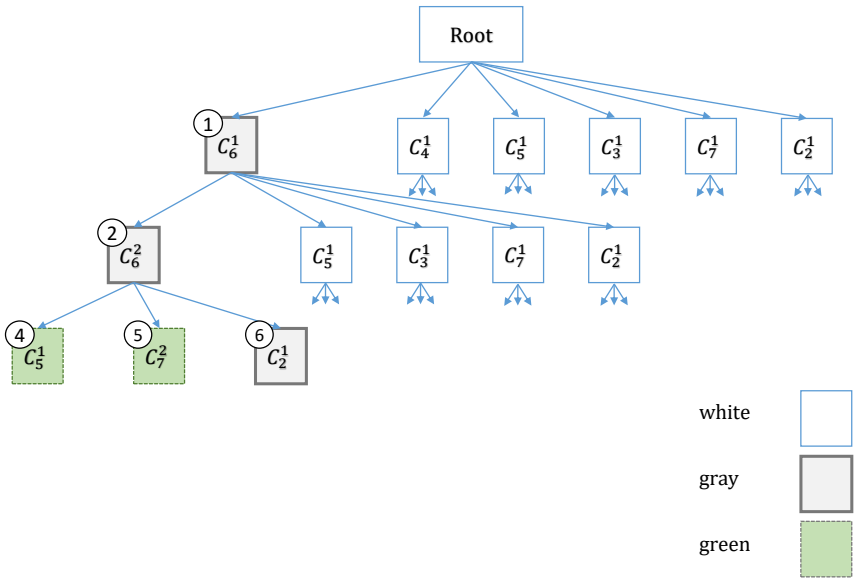

Figure 2: Branching scheme 1

Then among the children of $\mathcal{N}_1 = \{C_6^1\}$*, node* $\mathcal{N}_2 = \{C_6^1, C_6^2\}$ *is branched first and so on.*

All gray nodes are those for which $\sum_{\mathbf{p}_l \in \bar{Y}^{\mathcal{N}}} \pi_l < \hat{\alpha}$*. All green nodes are those for which* $\sum_{\mathbf{p}_l \in \bar{Y}^{\mathcal{N}}} \pi_l = \hat{\alpha}$*. For example, consider the node* $\mathcal{N}_2 = \{C_6^1, C_6^2\}$*. For this node* $\bar{Y}^{\mathcal{N}_2} = \{\mathbf{p}^2, \mathbf{p}^7, \mathbf{p}^9, \mathbf{p}^{10}\}$ *and* $\sum_{\mathbf{p}^l \in \bar{Y}^{\mathcal{N}_2}} \pi_l = 0.3 < \hat{\alpha}$ *(where* $\hat{\alpha} = 0.4$*), therefore its background color is gray. For the node* $\mathcal{N}_5 = \{C_6^1, C_6^2, C_7^2\}$*,* $\bar{Y}^{\mathcal{N}_5} = \{\mathbf{p}^2, \mathbf{p}^5, \mathbf{p}^7, \mathbf{p}^9, \mathbf{p}^{10}\}$ *and* $\sum_{\mathbf{p}^l \in \bar{Y}^{\mathcal{N}_5}} \pi_l = 0.4 = \hat{\alpha}$*, therefore its background color is green. We also compute:*

$$S^{Y^{\mathcal{N}_2}} = (0, 0, 3, 5, 0, 9, 9, 14, 19, 22) \rightarrow \text{UB}^{\mathcal{N}_5} = 22$$
$$S^{Y^{\mathcal{N}_5}} = (0, 0, 0, 8, 3, 12, 11, 12, 18, 21) \rightarrow \text{UB}^{\mathcal{N}_6} = 21$$

Some of the nodes can be dominated by computing a lower bound. In each node $\mathcal{N}_s$, we compute a lower bound, that is denoted by $\text{LB}^{\mathcal{N}_s}$. This lower bound is computed as explained in the following steps:

1. We construct $\hat{\mathcal{N}}_s$ which represents an extremely pessimistic case. $\hat{\mathcal{N}}_s$ initially contains all chains in $\mathcal{N}_s$.

2. Let $C_i^k = \Theta(\mathcal{N}_s)$. Add all chains $C_i^{k_1}$ with $k_1 = k+1, ..., \kappa_i$ to $\hat{\mathcal{N}}_s$ where κ_i is the largest integer that satisfies the following condition:

$$\sum_{\mathbf{p}_l \in \bar{Y}^{\mathcal{N}_s \cup \{C_i^{k+1}\} \cup ... \cup \{C_i^{\kappa_i}\}}} \pi_l \leq \hat{\alpha}.$$

3. Let activity i be the activity associated with the current node. For each activity $i_1 \in N$ that is positioned after activity i in the AL, add all chains $C_{i_1}^{k_1}$ with $k_1 = 1, ..., \kappa_{i_1}$ to $\hat{\mathcal{N}}_s$ where κ_{i_1} is the largest integer that satisfies the following condition:

$$\sum_{\mathbf{p}_l \in \bar{Y}^{\mathcal{N}_s \cup \{C_{i_1}^1\} \cup ... \cup \{C_{i_1}^{\kappa_{i_1}}\}}} \pi_l \leq \hat{\alpha}.$$

4. $\text{LB}^{\mathcal{N}_s} = S_{n+1}^{Y^{\hat{\mathcal{N}}_s}}$.

Every node $\mathcal{N}_s$ in our B&B tree is dominated if $\text{LB}^{\mathcal{N}_s} \geq \text{UB}^*$. Notice that this dominance rule is not considered in the example tree presented in Figure 2.

Example. *Consider node* $\mathcal{N}_2 = \{C_6^1, C_6^2\}$ *in Figure 2. We have:* $\bar{Y}^{\mathcal{N}_2} = \{\mathbf{p}^2, \mathbf{p}^7, \mathbf{p}^9, \mathbf{p}^{10}\}$ *and* $\sum_{\mathbf{p}^l \in \bar{Y}^{\mathcal{N}_2}} \pi_l = 0.3 < \hat{\alpha}$. *We compute* $\kappa_6 = 2, \kappa_4 = 0, \kappa_5 = 1, \kappa_3 = 1, \kappa_7 = 2$ *and* $\kappa_2 = 1$. *Therefore,* $\hat{\mathcal{N}}_2 = \{C_6^1, C_6^2, C_5^1, C_3^1, C_7^1, C_7^2, C_2^1\}$ *and* $\bar{Y}^{\hat{\mathcal{N}}_2} = \{\mathbf{p}^2, \mathbf{p}^5, \mathbf{p}^7, \mathbf{p}^8, \mathbf{p}^9, \mathbf{p}^{10}\}$. *We also compute:*

$$S^{Y^{\mathcal{N}_2}} = (0, 0, 3, 5, 0, 9, 9, 14, 19, 22) \rightarrow \text{UB}^{\mathcal{N}_2} = 22$$
$$S^{Y^{\hat{\mathcal{N}}_2}} = (0, 0, 3, 5, 0, 9, 9, 13, 17, 20) \rightarrow \text{LB}^{\mathcal{N}_2} = 20$$

5.1.3 Branching scheme 2

Similarly to branching scheme 1, branching scheme 2 also branches over chains of realizations. Each node is associated with a set of excluded chains. Without loss of generality, we use the same notation $\mathcal{N}_s$ to represent the sth node in the tree. Branching scheme 2 differs from branching scheme 1 in two major ways. Firstly, in each node of branching scheme 2, a set of target chains can be excluded (note that this set can be an empty set) instead of one single target chain. This set of target chains is denoted by $\Omega(\mathcal{N}_s)$. Secondly, each level of the tree is associated with a certain activity.

The branching starts with the root node. The root node is branched into a number of child nodes associated with the first activity in the AL,

each corresponding with the exclusion of a certain set of chains (remember that this set can be an empty set). Let us assume that activity i is the first activity in the AL. The first child node is associated with the set of chains $\{C_i^1, ..., C_i^{\zeta_i}\}$, the second child node is associated with exclusion of the set of chains $\{C_i^1, ..., C_i^{\zeta_i - 1}\}$, and so on. Finally, the last node is associated with the exclusion of no chain. Each of these child nodes is then branched into its own children, which are associated with the second activity in the AL, and so on. Backtracking happens in a node if all its children have already been visited, if its set of eligible chains is an empty set of if the exclusion or its target chains is infeasible.

For branching scheme 2, $\text{LB}^{\mathcal{N}_s}$ is computed based on the following steps:

1. We construct $\hat{\mathcal{N}}_s$ which represents an extremely pessimistic case. $\hat{\mathcal{N}}_s$ initially contains all chains in $\mathcal{N}_s$.

2. Let activity i be the activity associated with the current node. For each activity $i_1 \in N$ that is positioned after activity i in the AL, add all chains $C_{i_1}^{k_1}$ with $k_1 = 1, ..., \kappa_{i_1}$ to $\hat{\mathcal{N}}_s$ where κ_{i_1} is the largest integer that satisfies the following condition:

$$\sum_{\mathbf{p}_l \in \bar{Y}^{\mathcal{N}_s \cup \{C_{i_1}^1\} \cup ... \cup \{C_{i_1}^{\kappa_{i_1}}\}}} \pi_l \leq \hat{\alpha}.$$

3. $\text{LB}^{\mathcal{N}_s} = S_{n+1}^{Y^{\hat{\mathcal{N}}_s}}$.

Similarly to branching scheme 1, every node $\mathcal{N}_s$ in our B&B tree is dominated if $\text{LB}^{\mathcal{N}_s} \geq \text{UB}^*$. Beware that this dominance rule is also considered in the tree presented in Figure 3.

Example. *Figure 3 depicts the B&B tree where branching scheme 2 is used in a best-first mode. Each node is represented by a square. The root node is branched into three nodes:* $\mathcal{N}_1$, $\mathcal{N}_2$ *and* $\mathcal{N}_3$. *Among these three nodes, node* $\mathcal{N}_3$ *is branched first since its lower bound is smaller than that of the other two nodes.* $\mathcal{N}_3$ *is branched into two nodes:* $\mathcal{N}_4$ *and* $\mathcal{N}_5$. *Node* $\mathcal{N}_4$ *whose lower bound is larger than the best upper bound (*UB^**) found so far is eliminated from the three, whereas* $\mathcal{N}_5$ *is branched into its children (*$\mathcal{N}_6$, $\mathcal{N}_7$ *and* $\mathcal{N}_8$*). The next node to be branched is* $\mathcal{N}_1$ *because its lower bound is smaller than the lower bounds of all other unbranched nodes. The branching continues with the same logic until no unbranched node with a lower bound smaller than* UB^* *exists in the tree.*

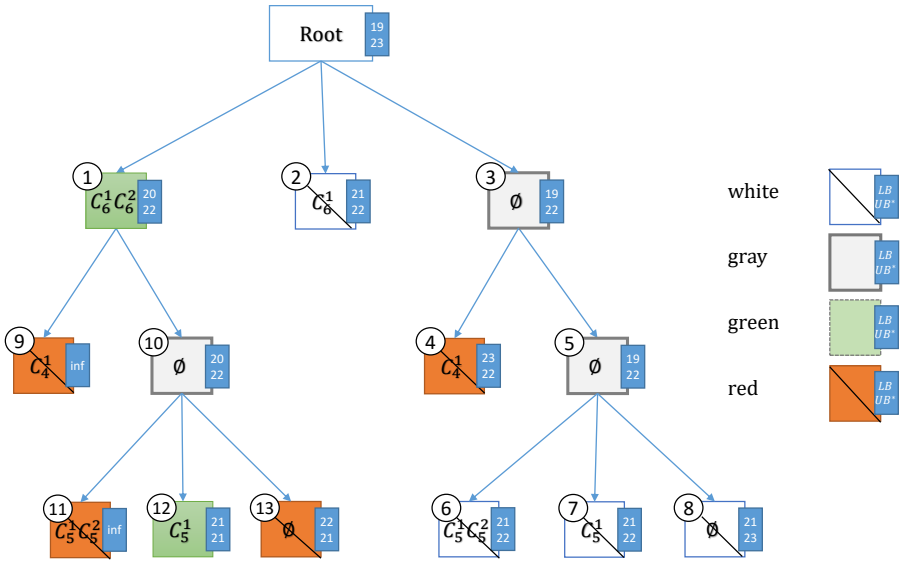

Figure 3: Branching scheme 2

All white nodes with a line over them are those that are left unbranched in the tree after the branching stopped. All gray nodes are those for which $\sum_{\mathbf{p}_l \in \bar{Y}^{\mathcal{N}}} \pi_l < \hat{\alpha}$ *and* $\mathrm{LB}^{\mathcal{N}} < \mathrm{UB}^* \leq \mathrm{UB}^{\mathcal{N}}$ *and thus they are branched from and not eliminated. All green nodes are those for which* $\sum_{\mathbf{p}_l \in \bar{Y}^{\mathcal{N}}} \pi_l \leq \hat{\alpha}$ *and* $\mathrm{UB}^{\mathcal{N}} < \mathrm{UB}^*$. *All red nodes with a line over them are those for which* $\sum_{\mathbf{p}_l \in \bar{Y}^{\mathcal{N}}} \pi_l > \hat{\alpha}$ *or* $\mathrm{LB}^{\mathcal{N}} \geq \mathrm{UB}^*$ *and hence they are eliminated.*

5.2 Improvements by hashing and listing

Although $\mathrm{LB}^{\mathcal{N}_s}$ is a valid lower bound and thus can be used to prune our B&B tree, its computation can be costly since it requires calling the optimization oracle $\mathcal{O}(.)$. Therefore, one might be interested in finding a computationally much cheaper lower bounding approach. Assume $\mathbf{p}^{\hat{Y}}$ is a vector of durations for which $\mathcal{O}(\mathbf{p}^{\hat{Y}})$ has already been solved and $S^{\hat{Y}}$ is its resulting optimal solution. If $\mathbf{p}^{\hat{Y}} \leq \mathbf{p}^{Y^{\hat{\mathcal{N}}_s}}$, then $S^{\hat{Y}}_{n+1} \leq \mathrm{LB}^{\mathcal{N}_s}$. In our B&B algorithm, we call the optimization oracle $\mathcal{O}(\mathbf{p}^{Y^{\hat{\mathcal{N}}_s}})$ only if for each $\mathbf{p}^{\hat{Y}} \leq \mathbf{p}^{Y^{\hat{\mathcal{N}}_s}}$, the

inequality $S_{n+1}^{\hat{Y}} < \text{UB}^*$ is satisfied. Otherwise, the node $\mathcal{N}_s$ is dominated.

In our algorithm, every time the oracle $\mathcal{O}(\mathbf{p}^Y)$ is solved, we store the pair $(\mathbf{p}^Y, S^Y)$ both in a hash table and in a linked list. The hash table is used to avoid calling $\mathcal{O}(\mathbf{p}^Y)$ more than once for the nodes with a common set of excluded realizations whereas the linked list is used for the lower bound computation. Before calling the optimization oracle to compute the lower bound, we check all pairs $(\mathbf{p}^Y, S^Y)$ in the linked list and ensure that no pair in the linked list is sufficient to conclude the domination of the node. The members of the linked list are constantly ordered by the number of times they successfully caused a domination.

6 Computational results

In this section, we report computational results for our B&B algorithm. We also compare the performance of our B&B algorithm with the given MILP formulation in Section 4 and the branch-and-cut algorithm proposed by Lamas and Demeulemeester (2016). To implement the B&B algorithm, the MILP formulation and the B&C algorithm, Visual C++ 2010 and CPlex 12.5.1 were used. All computational results were obtained on a computer with Intel(R) Xeon(R) CPU E5-2699 v3 2.30 GHz (2 processors, 36 cores), 256GB of RAM and running under Windows Server 2012 R2. It is worth mentioning that in our experiments, at each time instance, 32 problem instances ran in parallel such that each instance was using only one thread (core). The remaining four cores were deliberately kept idle to deal with any possible overhead tasks and thus refrain such tasks from significantly influencing the results in our experiments.

We chose a memory limit of 10 GB and a time limit of one hour to solve each instance of the problem using any of the methods. The B&B method and the B&C method usually required about a few hundreds MB of RAM (note that the required memory to solve the B&C algorithm is significantly larger than that of our B&B algorithm). It is worth mentioning that, in our experiments, these two algorithms never exceeded the memory limit. The MILP formulation, on the other hand, requires a larger amount of memory which is often still less than 10GB. This method exceeded the memory limit only once in our experiments. If an instance is solved within the time and memory limits, it is labeled *solved*, and otherwise *unsolved*.

6.1 Instance generation

All methods are tested on a set of instances that are composed of the PSPLIB instances. Only instances with 30 non-dummy activities are considered in this experiment. PSPLIB is a class of instances for the deterministic RCPSP (Kolisch and Sprecher, 1997), thus they need to be modified to suit our problem. The following modifications are applied on this set of instances: the activity durations $\tilde{p}_i$ for each non-dummy activity i follow a discretized beta distribution with shape parameters 2 and 5 that is mapped over the interval $[0.75\hat{p}_i, 1.625\hat{p}_i]$ where $\hat{p}_i$ is the duration of activity i that is given in the original instance. In order to reduce the number of experiments, we only consider the instances from the set J30 of PSPLIB with the following filename syntax: J30X_1 ($X = 1, ..., 48$). The random generator's seed for the instance obtained from J30X_1 equals X.

The size of the set $\mathcal{P}$ can be extremely large. Any algorithm (including MILP solvers) that solves the problem might not be computationally tractable if $\mathcal{P}$ is large. Therefore, we apply a sample average approximation technique to deal with the problem. We generate several sets of realizations with different sizes ($m = 100, 200, 400, 800$ or 1600) as explained in Lamas and Demeulemeester (2016). We select $1 - \hat{\alpha}$ from the set $\{0.99, 0.95, 0.90, 0.80\}$. For each combination of $(X, m, (1 - \hat{\alpha}))$, an instance results and thus the total number of instances is $48 \times 5 \times 4 = 960$.

6.2 Overall results

We run our B&B algorithm on the set of instances described in Section 6.1 using different branching schemes and different priority rules. We report the overall results in Table 5. In separate experiments, we evaluate the performance of our B&B algorithm for each combination of a branching scheme and a priority rule. Branching schemes that are used in these experiments are branching scheme 1 (BS1) in depth-first mode, BS1 in a best-first mode and branching scheme 2 (BS2) in a best-first mode. Notice that we deliberately decide not to include BS2 in a depth-first mode in our experiments since the order in which the sets of excluded chains are evaluated in BS2 in a depth-first mode highly resembles that in BS1 in a depth-first mode. Also, preliminary results indicate that the required CPU time, the number of nodes and the number of oracle calls for these two approaches are extremely close. We use all priority rules described in Section 5.1.1 in these experiments.

Among all branching schemes, branching scheme 1 in a depth-first mode

Table 5 Average CPU times (in seconds) and number of solved instances within the time limit (out of 960) for different choices of priority rules and different branching schemes.

Priority rule	BS1		BS2
	Depth-first	Best-first	Best-first
R_1	339.60 (897)	408.98 (884)	422.01 (878)
R_2	347.49 (895)	394.20 (885)	389.61 (883)
R_3	**178.81 (925)**	213.08 (919)	207.64 (920)
R_4	211.58 (917)	233.99 (914)	238.86 (910)
R_5	374.06 (887)	447.94 (874)	469.47 (866)
R_6	495.57 (861)	624.88 (831)	656.77 (820)
R_7	540.97 (855)	611.26 (838)	620.38 (831)
R_8	536.24 (856)	608.88 (838)	615.14 (836)

clearly performs the best. This better performance can be justified by expressing the importance of tight upper bounds. Obviously, a tight upper bound in the early stages of searching the tree can help pruning the low-quality branches. In a best-depth mode, such a tight upper bound is often obtained very late.

Our B&B algorithm performs best when R_3 is used to construct the activity list. This suggests that the total slack is the most important criterion, the number of chains is the second important criterion and the influence factor is the least important criterion. It also suggests that the choice of priority rule significantly influences the performance of our B&B algorithm.

The comparison between different settings in Table 5 is not very clear because the number of solved instances varies for different settings. In order to provide a better comparison between settings, each pair must be separately compared in more detail. In such a comparison, one should only consider the instances that are solved to optimality in the settings under question. In Table 6, we compare two different settings, namely the combination of BS1 and R_1 (also denoted by the pair $(\text{BS1}, R_1)$) in a depth-first mode and $(\text{BS1}, R_3)$ also in a depth-first mode. Let $\text{CPU}_I(\text{BS1}, R_1)$ be the CPU time required to optimally solve instance I using the former setting and $\text{CPU}_I(\text{BS1}, R_3)$ be the CPU time required to optimally solve instance I using the latter setting. We compute the average percentage deviation of the required CPU time using $(\text{BS1}, R_3)$ in a depth-first mode from the

Table 6 The average percentage deviation of the required CPU time using $(\text{BS1}, R_3)$ in a depth-first mode from the required CPU time using $(\text{BS1}, R_1)$ in a depth-first mode (in percentage) and the number of instances solved in both settings (out of 48) for different choices of $1-\hat{\alpha}$ and m

$1-\hat{\alpha}$	m				
	100	200	400	800	1600
0.99	-8.91 (48)	-7.21 (48)	-39.37 (48)	-32.65 (48)	-47.94 (48)
0.95	-42.27 (48)	-53.72(47)	-61.07 (47)	-65.85 (46)	-78.26 (44)
0.90	-48.64 (46)	-70.54 (47)	-71.63 (46)	-65.44 (44)	-80.60 (43)
0.80	-69.19 (45)	-83.97 (45)	-74.74 (40)	-85.25 (37)	-94.18 (30)

required CPU time using $(\text{BS1}, R_1)$ in a depth-first mode as follows:

$$\text{avg}\left\{\frac{\text{CPU}_I(\text{BS1}, R_3) - \text{CPU}_I(\text{BS1}, R_1)}{\text{CPU}_I(\text{BS1}, R_1)} \times 100\%\right\}$$

According to the results that are presented in Table 6, for every choice of $1-\hat{\alpha}$ and m, $(\text{BS1}, R_3)$ performs better than $(\text{BS1}, R_1)$ (negative average percent deviation). We notice that by increasing the number of realizations (m) and also by decreasing the confidence level $(1-\hat{\alpha})$, the associated average percent deviation decreases. In other words, by increasing the number of realizations and also by decreasing the confidence level, the difference between the performances of $(\text{BS1}, R_3)$ and $(\text{BS1}, R_1)$ becomes more significant. For instance, when $1-\hat{\alpha} = 0.95$ and $m = 200$, $(\text{BS1}, R_3)$ performs about two times faster than $(\text{BS1}, R_1)$ whereas when $1-\hat{\alpha} = 0.80$ and $m = 1600$, $(\text{BS1}, R_3)$ performs about 17 times faster than $(\text{BS1}, R_1)$.

Among all combinations, $(\text{BS1}, R_3)$ in a depth-first mode performs the best. In the remainder of this paper, we only report the results of our B&B for this combination and the results associated with other combinations are ignored. Note that in the following subsections, for the sake of simplicity, we mention no setting and instead we simply use the notion 'our B&B'.

6.3 Detailed results

In Table 7, for each pair $((1-\hat{\alpha}), m)$, we report the number of instances that are solved within the time limit (Solved), the average and maximum CPU times (avg(CPU) and max(CPU)), the average number of chains (avg($|\mathbf{C}^E|$)), the average number of nodes visited in the tree (avg(NN)) and the average number of times the optimization oracle is called (avg(OC)).

We observe that by decreasing $1-\hat{\alpha}$ from 0.99 to 0.80 and/or by increasing the number of realizations (m), the average number of chains, the

Table 7 The detailed computational results for our B&B algorithm

$1-\hat{\alpha}$	m	Solved	CPU		avg($\lvert\mathbf{C}^E\rvert$)	avg(NN)	avg(OC)
			avg	max			
0.99	100	48	2.71	118.56	6.94	6.73	7.73
	200	48	3.29	133.55	10.38	16.33	14.40
	400	48	18.95	859.73	14.46	45.38	28.83
	800	48	22.76	943.68	17.56	136.98	66.63
	1600	48	72.04	2994.27	21.10	243.94	84.71
0.95	100	48	32.61	1360.92	22.19	113.96	71.73
	200	48	60.56	2611.33	26.02	327.60	161.38
	400	47	95.85	3600.00	30.35	1076.27	380.25
	800	46	171.39	3600.00	33.98	3031.90	825.19
	1600	46	206.83	3600.00	37.19	5484.21	1290.60
0.90	100	47	101.44	3600.00	31.75	732.35	358.17
	200	47	110.79	3600.00	36.42	1893.04	729.15
	400	46	184.42	3600.00	40.46	5725.63	1586.52
	800	46	263.40	3600.00	44.27	18093.48	4176.90
	1600	45	296.78	3600.00	48.19	26253.33	5392.23
0.80	100	46	258.06	3600.00	44.42	6028.85	2270.23
	200	46	212.91	3600.00	49.08	17682.98	4882.92
	400	43	402.35	3600.00	53.58	39607.83	9144.67
	800	42	509.95	3600.00	57.79	136760.71	24427.54
	1600	42	549.09	3600.00	61.98	156366.33	27565.60

average CPU times, the average number of nodes and the average number of oracle calls are often increased whereas the average number of solved instances is decreased or remains unchanged.

We report that the average number of oracle calls is increased almost linearly by increasing the number of realizations, which is the main strength of this approach. However, the average number of oracle calls is increased exponentially related to an increase in the number of chains (see Figure 4).

6.3.1 Quality of the lower bound

One of the main features of our B&B algorithm is the lower bound computation. To show the strength of the proposed lower bound, we report the average deviation of the lower bound in the root node from the objective value of the best found (optimal) solution in Table 8. This average deviation is computed as follows:

$$\text{avg}\left\{\frac{\text{LB}^{\mathcal{N}_0} - UB^*}{UB^*} \times 100\%\right\}$$

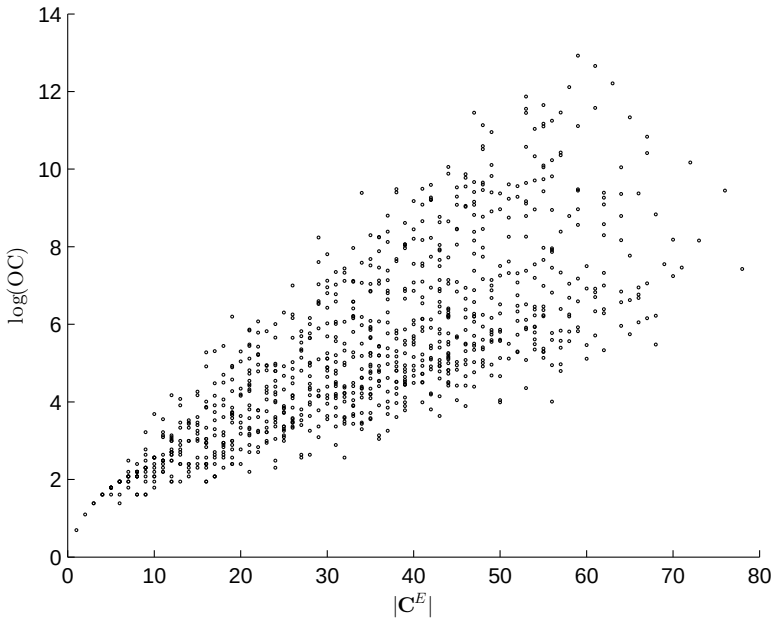

Figure 4: Logarithm of number of oracle calls versus number of chains

Table 8 The average percent deviation between the lower bound and the objective value of the best found (or optimal) solution for different choices of $1 - \hat{\alpha}$ and m.

$1 - \hat{\alpha}$	m				
	100	200	400	800	1600
0.99	-2.23	-2.63	-3.16	-3.26	-3.17
0.95	-5.86	-5.71	-6.28	-6.12	-6.35
0.90	-7.68	-8.39	-8.15	-8.25	-8.25
0.80	-10.69	-11.31	-11.06	-11.44	-11.67

Notice that the computation of the lower bound in the root node is exactly the same for the two branching schemes. The results in Table 8 suggest that by decreasing the confidence level and by increasing the number of realizations, the quality of the lower bound is generally decreased.

6.3.2 Impacts of hashing and listing

In Section 5.2, we discussed two improvement techniques, namely exploiting a hash table (HT) and a linked list (LL). Table 9 demonstrates their

Table 9 The effect of implementing the hash table (HT) and/or the linked list (LL) on our B&B algorithm

Setting	Solved (out of 960)	avg(CPU)	avg(OC)
B&B	925	178.81	4173.27
B&B - HT	925	180.81	4514.47
B&B - LL	913	239.61	19483.30
B&B - HT - LL	913	241.19	19686.32

effects on the performance of our B&B algorithm. The first row represents the setting in which both the hash table and the linked list are exploited. The second and third row represent the settings in which either of the two improving techniques is not exploited. Finally, the last low represents the setting in which none of them is exploited. Based on these results, we notice that implementing the hash table slightly improves the performance of our B&B algorithm whereas the improvement resulting from the implementation of the linked list is significant.

6.3.3 Results for instances with large number of realizations

Readers may be interested in the performance of our B&B algorithm when larger sets of realizations are considered. In this part, we present the results of an additional set of instances. This set is generated similarly to the set of instances introduced in Section 6.1, except that $m = 3200, 6400, 12800, 25600$ or 51200. Table 10 reports the average CPU times and the numbers of instances solved for different choices of $1 - \hat{\alpha}$ and these large m values. We notice that our B&B algorithm can solve 35 (out of 48) instances even for the most difficult setting, *i.e.* the setting where $m = 51200$ and $1-\hat{\alpha} = 0.80$. The main reason of such a good performance lies behind the way the chains are introduced. Having a fixed range of integer numbers, the number of chains remains almost constant by increasing the number of realizations. Note that the number of oracle calls and as such the required CPU times are exponentially increased only by increasing the number of chains, as it has been shown in Figure 4. Therefore, because the number of chains remains almost constant by increasing the number of realizations, we conclude that increasing the number of realizations does not lead to a significant increase in CPU times.

Table 10 The average CPU time and the number of instances solved (out of 48) for different choices of $1-\hat{\alpha}$ and large m values

$1-\hat{\alpha}$	m				
	3200	6400	12800	25600	51200
0.99	35.66 (48)	92.69 (48)	134.54 (47)	160.03 (47)	245.81 (45)
0.95	247.56 (45)	347.23 (44)	388.62 (45)	381.67 (44)	413.95 (44)
0.90	406.65 (44)	484.43 (44)	517.01 (42)	675.13 (41)	737.77 (40)
0.80	697.72 (40)	846.37 (39)	946.33 (38)	1029.67 (36)	1037.80 (35)

6.3.4 Results for instances with medium and high variances

The activity durations of the instances introduced in Section 6.1 are generated from the range $[0.75\hat{p}_i, 1.625\hat{p}_i]$ which represents a low variance. In most papers that are dealing with project scheduling where activity durations are stochastic, two wider ranges are also used: a range of $[0.5\hat{p}_i, 2.25\hat{p}_i]$ which represents a medium variance and a range of $[0.25\hat{p}_i, 2.875\hat{p}_i]$ which represents a high variance. We introduce two additional sets of instances that are generated similarly to the set of instances introduced in Section 6.1, except that the medium and the high variance ranges are used to generate the activity durations. Tables 11 and 12 show the detailed computational results for our B&B algorithm ran on instances with medium and high variances, respectively. As expected, by increasing the range (variance), the number of chains and as such the CPU times are increased whereas the number of solved instances is decreased. It is interesting to see that our B&B algorithm can also solve most of the instances with medium variance and a reasonably large number of instances with high variance to optimality within the time limit.

6.4 Comparison with other methods

We compare our B&B algorithm with the mathematical formulations proposed in Section 4 and the B&C algorithm proposed by Lamas and Demeulemeester (2016). For each pair $(1-\hat{\alpha}, m)$, Table 13 reports the number of solved instances within different time limits: 10 seconds (10s), 1 minute (1m), 10 minutes (10m) and 1 hour (1h). We observe that our B&B algorithm clearly outperforms the mathematical formulation and the B&C algorithm in all settings.

As we decrease $(1-\hat{\alpha})$, the number of solved instances is decreased for all three methods. The same behavior is noticed when we increase the number of realizations. Interestingly, we notice that, within 10 seconds, our B&B algorithm can solve more instances than the number of instances

Table 11 The detailed computational results for our B&B algorithm ran on instances with medium variances

$1-\hat{\alpha}$	m	Solved	CPU		avg($\lvert\mathbf{C}^E\rvert$)	avg(NN)	avg(OC)
			avg	max			
0.99	100	48	3.99	166.937	13.15	12.52	13.50
	200	48	22.70	1007.54	20.77	45.50	41.33
	400	48	89.79	3490.83	29.19	238.63	184.52
	800	47	120.67	3600.00	36.29	1274.45	659.21
	1600	46	249.27	3600.00	42.42	6458.27	2583.79
0.95	100	47	109.14	3600.00	42.06	835.25	628.56
	200	45	313.76	3600.00	52.75	5612.54	2950.17
	400	44	480.87	3600.00	62.13	54531.02	19719.38
	800	39	842.35	3600.00	69.25	218721.46	50665.90
	1600	36	1178.17	3600.00	75.50	399935.52	85520.31
0.90	100	45	313.66	3600.00	61.23	12494.23	6862.81
	200	42	570.21	3600.00	73.31	74333.40	27812.06
	400	34	1274.74	3600.00	82.67	324515.38	86804.75
	800	33	1294.26	3600.00	91.17	441609.19	92741.42
	1600	28	1582.64	3600.00	97.98	576703.73	115854.06
0.80	100	38	903.09	3600.00	85.13	175219.60	67898.67
	200	34	1266.76	3600.00	97.88	330257.13	98129.88
	400	29	1591.19	3600.00	108.33	522442.42	126655.79
	800	27	1700.57	3600.00	116.96	653758.15	128094.58
	1600	24	1905.06	3600.00	124.79	935444.83	167067.04

the mathematical formulation can solve within one hour or the number of instances the B&C algorithm can solve within 10 minutes.

As it is also clear in the table, B&C is generally performing better than the MILP formulation. However, when $1-\hat{\alpha} \geq 0.90$ and $m \geq 800$ the MILP formulation sometimes performs better, specially for the smaller time limits. This somewhat unexpected result might be because of the pre-processing steps in the B&C algorithm.

7 Discussion: general CCP problem

Although our proposed B&B algorithm is applied to solve the SAA-RCPSP, it can be used to solve any CCP problem with discrete random rhs vector provided that an optimal solution methodology exists for its deterministic counterpart. For instance, consider the classical transportation problem with random discrete demand vector. One could use the same B&B algorithm proposed in Section 5 to solve the chance-constrained version of the classical transportation problem. In this case, $\mathcal{O}(.)$ should be an oracle that optimally solves the deterministic transportation problem.

Table 12 The detailed computational result for our B&B algorithm ran on instances with high variances.

$1-\hat{\alpha}$	m	Solved	CPU		avg($\lvert\mathbf{C}^E\rvert$)	avg(NN)	avg(OC)
			avg	max			
0.99	100	48	11.08	502.98	15.31	14.42	15.35
	200	48	29.25	1189.15	26.15	68.00	65.52
	400	47	141.76	3600.00	39.67	576.04	495.94
	800	45	319.38	3600.00	50.75	4691.56	2697.77
	1600	43	442.91	3600.00	62.40	26714.71	10894.48
0.95	100	47	131.68	3600.00	53.27	1199.21	965.15
	200	43	416.64	3600.00	70.67	12984.54	7192.79
	400	36	957.54	3600.00	87.29	162026.98	63291.98
	800	31	1404.69	3600.00	100.33	357048.48	104380.56
	1600	27	1632.23	3600.00	112.33	529853.10	126483.63
0.90	100	43	470.28	3600.00	81.10	20620.33	13079.69
	200	36	1053.60	3600.00	100.50	191257.79	82103.69
	400	27	1769.10	3600.00	117.21	427891.35	141403.94
	800	25	1880.22	3600.00	129.98	615474.25	158701.44
	1600	21	2274.00	3600.00	142.96	851616.31	183599.63
0.80	100	32	1408.39	3600.00	118.40	227325.42	109373.13
	200	22	2090.18	3600.00	138.65	462194.48	170729.17
	400	18	2440.46	3600.00	155.69	645457.58	195175.23
	800	16	2653.74	3600.00	169.06	910299.63	216549.46
	1600	13	2892.69	3600.00	181.88	1111028.08	232467.46

8 Summary and conclusion

In this paper, we propose a novel B&B algorithm that solves the SAA-RCPSP in a much more efficient manner than the methods that are already existing in the literature. The goal in SAA-RCPSP is to select a subset of realizations, for which the optimal solution must be feasible, such that the resulting confidence level is at least $(1-\hat{\alpha})$. Instead of branching over realizations, we branch over chains of realizations and thus the complexity of the method is a function of the number of chains rather than a function of the number of realizations. If the activity realizations are discrete, then the number of chains usually is increased slightly by increasing the number of the realizations.

In our experiments, we tested different priority rules for activities, based on which the B&B tree is constructed. We noticed that, among several sorting criteria, sorting activities based on smaller total slack times significantly improves the performance of our B&B algorithm.

We ran our B&B algorithm together with a MILP formulation and a B&C algorithm on benchmark instances of size 30 activities. The B&B

Table 13 The number of instances solved to optimality for different time limits (10 seconds, 1 minute, 10 minutes and 1 hour), different methods and different choices of $1-\hat{\alpha}$ and large m values

$1-\hat{\alpha}$	m	B&B				SAA-RCPSP-MILP2				B&C			
		10s	1m	10m	1h	10s	1m	10m	1h	10s	1m	10m	1h
0.99	100	47	47	48	48	26	30	33	38	39	44	47	47
	200	47	47	48	48	26	30	32	35	38	43	47	47
	400	46	47	47	48	24	29	32	36	38	40	46	46
	800	43	46	47	48	22	28	31	34	33	39	43	46
	1600	43	45	47	48	19	30	32	35	28	37	42	46
0.95	100	44	46	47	48	21	30	30	37	35	39	43	46
	200	42	45	47	48	19	30	31	36	30	39	42	44
	400	41	43	47	47	18	29	32	36	24	34	40	43
	800	42	43	45	46	9	24	32	37	16	27	38	40
	1600	41	43	44	46	6	25	29	32	6	22	31	39
0.90	100	41	44	46	47	17	28	31	37	28	38	42	44
	200	42	42	46	47	10	25	31	35	23	32	38	42
	400	40	43	45	46	8	23	31	33	12	26	38	40
	800	40	41	43	46	5	19	31	32	9	19	29	39
	1600	36	41	43	45	0	10	26	31	0	7	23	32
0.80	100	39	41	44	46	9	22	30	32	18	29	40	42
	200	33	41	44	46	7	22	30	31	10	25	35	39
	400	32	38	42	43	5	13	28	29	4	15	27	36
	800	30	35	40	42	1	5	23	28	0	8	21	29
	1600	31	33	40	42	0	4	10	24	0	0	13	19

algorithm outperforms both the MILP formulation and the B&C algorithm both in terms of computational times and the number of solved instances within the time limit.

References

Artigues, C., Michelon, P., and Reusser, S. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249 – 267, 2003.

Artigues, C., Demassey, S., and Neron, E. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*. Wiley, 2008.

Ashtiani, B., Leus, R., and Aryanezhad, M. B. New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits ofpre-processing. *Journal of Scheduling*, 14(2):157–171, 2011.

Aytug, H., Lawley, M. A., McKay, K., Mohan, S., and Uzsoy, R. Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161(1):86–110, 2005.

Blazewicz, J., Lenstra, J. K., and Rinnooy Kan, A. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

Creemers, S. Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. *Journal of Scheduling*, 18 (3):263–273, 2015.

Davari, M. and Demeulemeester, E. The proactive and reactive resource-constrained project scheduling problem. Technical Report KBI_1613, KU Leuven, 2016.

Deblaere, F., Demeulemeester, E., and Herroelen, W. Reactive scheduling in the multi-mode RCPSP. *Computers & Operations Research*, 38(1):63–74, 2011a.

Deblaere, F., Demeulemeester, E., and Herroelen, W. Proactive policies for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research*, 214(2):308–316, 2011b.

Demeulemeester, E. and Herroelen, W. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38(12):pp. 1803–1818, 1992.

Demeulemeester, E. and Herroelen, W. An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem. *European Journal of Operational Research*, 90(2):334–348, 1996.

Demeulemeester, E. and Herroelen, W. New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43(11):pp. 1485–1492, 1997.

Demeulemeester, E. and Herroelen, W. *Project Scheduling: A Research Handbook*. Kluwer Academic Publisher, 2002.

Demeulemeester, E. and Herroelen, W. Robust project scheduling. *Foundations and Trends in Technology, Information and Operations Management*, 3:201–376, 2011.

Herroelen, W. Project scheduling: Theory and practice. *Production and Operations Management*, 14(4):413–432, 2005.

Herroelen, W. and Leus, R. The construction of stable project baseline schedules. *European Journal of Operational Research*, 156(3):550–565, 2004.

Klein, R. *Scheduling of Resource Constrained Projects*. Kluwer Academic Publisher, 2000.

Kolisch, R. and Sprecher, A. Psplib - a project scheduling problem library. *European Journal of Operational Research*, 96(1):205–216, 1997.

Koné, O., Artigues, C., Lopez, P., and Mongeau, M. Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, 38(1):3 – 13, 2011.

Küçükyavuz, S. On mixing sets arising in chance-constrained programming. *Mathematical Programming*, 132(1):31–56, 2012.

Lamas, P. and Demeulemeester, E. A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling*, 19(4):409–428, 2016.

Leus, R. *The generation of stable project plans*. PhD thesis, KU Leuven, 2003.

Leus, R. and Herroelen, W. The complexity of machine scheduling for stability with a single disrupted job. *Operations Research Letters*, 33(2): 151–156, 2005.

Luedtke, J. and Ahmed, S. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19 (2):674–699, 2008.

Luedtke, J., Ahmed, S., and Nemhauser, G. L. An integer programming approach for linear programs with probabilistic constraints. *Mathematical Programming*, 122(2):247–272, 2010.

Mehta, S. V. and Uzsoy, R. M. Predictable scheduling of a job shop subject to breakdowns. *Robotics and Automation, IEEE Transactions on*, 14(3): 365–378, 1998.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., and Bianco, L. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44(5):714–729, 1998.

Möhring, R. and Radermacher, F. Introduction to stochastic scheduling problems. In Neumann, K. and Pallaschke, D., editors, *Contributions to Operations Research*, volume 240 of *Lecture Notes in Economics and Mathematical Systems*, pages 72–130. Springer Berlin Heidelberg, 1985.

Möhring, R., Radermacher, F., and Weiss, G. Stochastic scheduling problems I - general strategies. *Zeitschrift für Operations Research*, 28(7): 193–260, 1984.

Möhring, R., Radermacher, F., and Weiss, G. Stochastic scheduling problems II - set strategies. *Zeitschrift für Operations Research*, 29(3):65–104, 1985.

Prékopa, A. Dual method for the solution of a one-stage stochastic programming problem with random rhs obeying a discrete probability distribution. *Zeitschrift für Operations Research*, 34(6):441–461, 1990.

Rostami, S., Creemers, S., and Leus, R. New benchmark results for the stochastic resource-constrained project scheduling problem. Technical report, KU Leuven, 2016.

Ruszczyński, A. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Mathematical Programming*, 93(2):195–215, 2002.

Sprecher, A. Scheduling resource-constrained projects competitively at modest memory requirements. *Management Science*, 46(5):710–723, 2000.

Stork, F. *Stochastic resource-constrained project scheduling*. PhD thesis, TU Berlin, 2001.

Stork, F. Branch-and-bound algorithms for stochastic resource-constrained project scheduling. Technical report, Technische Universitat Berlin, 2000.

Van de Vonder, S., Demeulemeester, E., Herroelen, W., and Leus, R. The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research*, 44(2):215–236, 2006a.

Van de Vonder, S., Demeulemeester, E., Leus, R., and Herroelen, W. *Perspectives in Modern Project Scheduling*, chapter Proactive-Reactive Project Scheduling Trade-Offs and Procedures, pages 25–51. Springer US, Boston, MA, 2006b.

Van de Vonder, S., Demeulemeester, E., and Herroelen, W. Proactive heuristic procedures for robust project scheduling: An experimental analysis. *European Journal of Operational Research*, 189(3):723–733, 2008.

FACULTY OF ECONOMICS AND BUSINESS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be